

UNIwersytet ZIELONOGÓRSKI	WYKONALI:	GRUPA		OCENA:
Rozpoznawanie obrazów				
ĆWICZENIE NUMER: 9	TEMAT: Algorytm k-średnich	DATA WYKONANIA:	DATA ODDANIA:	PODPIS:

1. Cel ćwiczenia

Celem ćwiczenia jest zaznajomienie się ze skutecznością klasyfikowania obiektów przy pomocy algorytmu „k-średnich”. Zadanie nasze będzie polegało na rozpoznawaniu wcześniej przygotowanych znaków alfabetu za pomocą algorytmu k-średnich a na koniec porównany wyniki z metodą k-najbliższych sąsiadów z poprzednich zajęć.

2. Zbudowanie klasyfikatora algorytmu k-średnich.

Załóżmy że posiadamy 4 skupiska różniące się od siebie zarówno ilością obiektów jak i samymi obiektami, przy czym każde skupienie składa się z jednakowej grupy elementów. Postaramy się teraz zbudować klasyfikator w oparciu o metodę k-średnich, który przy przypadkowo wylosowanych centrach, będzie podejmował próbę przypisania każdego centrum do określonego skupiska. Dodam jeszcze tylko iż skupiska powinny się maksymalnie różnić od siebie, aby centra zostały sklasyfikowane prawidłowo.

W przeciwnym przypadku może zaistnieć sytuacja w której centrum w kolejnych krokach algorytmu będzie się wahać pomiędzy jednym a drugim skupiskiem.

Skorzystaliliśmy z metody off-line algorytmu k-średnich.

Program wraz z algorytmem zaimplementowanym w środowisku MATLAB zamieszczamy poniżej.

```
macierz=zeros(100);
```

```
tab=zeros(64,4);
centra=zeros(4,2);
```

```
for i=1:13, x=round(rand()*32)+7; y=round(rand()*27)+4; macierz(y,x)=1; end
for i=1:14, x=70+round(rand()*25); y=round(rand()*23)+1; macierz(y,x)=2; end
for i=1:20, x=round(rand()*29)+10; y=round(rand()*23)+75; macierz(y,x)=3; end
for i=1:17, x=round(rand()*25)+75; y=round(rand()*25)+75; macierz(y,x)=4; end
for i=1:4, x=round(rand()*99)+1; y=round(rand()*99)+1; centra(i,2)=x; centra(i,1)=y; end
```

```
x1=0; y1=0; x2=0; y2=0; x3=0; y3=0; x4=0; y4=0;
```

```
for i=1:100
```

```
    for j=1:100
```

```
        if (macierz(j,i)==1) x1=x1+i; y1=y1+j; end
```

```
        if (macierz(j,i)==2) x2=x2+i; y2=y2+j; end
```

```
        if (macierz(j,i)==3) x3=x3+i; y3=y3+j; end
```

```
        if (macierz(j,i)==4) x4=x4+i; y4=y4+j; end
```

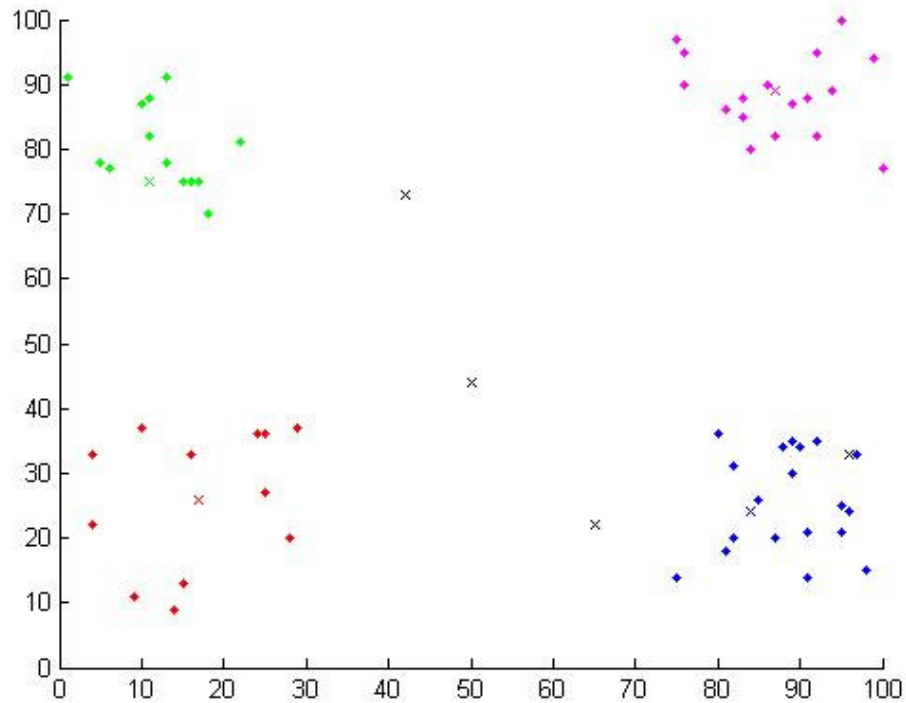
```
    end
```

```
end
```

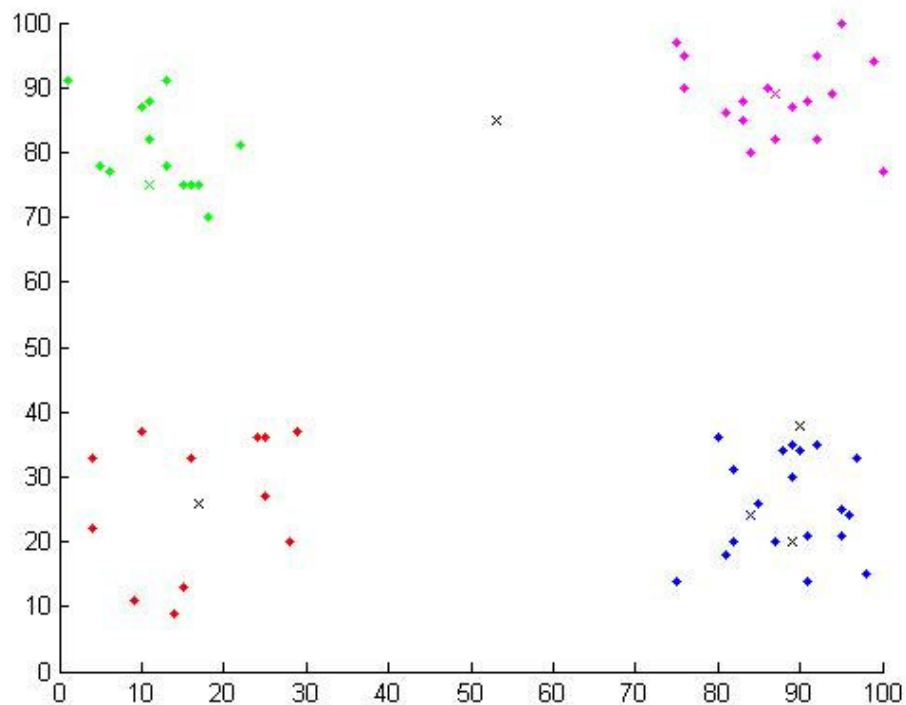
```
macierz(round(y1/13),round(x1/13))=9;
macierz(round(y2/14),round(x2/14))=10;
macierz(round(y3/20),round(x3/20))=11;
macierz(round(y4/17),round(x4/17))=12;
figure
for a=1:5,
% tablica center %%%%%%%%%%
L1=1;
for i=1:100
    for j=1:100
        if (macierz(j,i)==1 || macierz(j,i)==2 || macierz(j,i)==3 || macierz(j,i)==4),
            dl=zeros(4,1);
            for k=1:4, dl(k)=sqrt((i-centra(k,2))^2+(j-centra(k,1))^2); end
            [dl,index] = sort(dl);
            tab(L1,2)=i; tab(L1,1)=j; tab(L1,3)=index(1); tab(L1,4)=dl(1);
            L1=L1+1;
        end
    end
end
%clf;
figure
for i=1:100
    for j=1:100
        hold on;
        if (macierz(j,i)==1) plot(j,i,'-r'); end
        if (macierz(j,i)==2) plot(j,i,'-g'); end
        if (macierz(j,i)==3) plot(j,i,'-b'); end
        if (macierz(j,i)==4) plot(j,i,'-m'); end
        if (macierz(j,i)==9) plot(j,i,'x-r'); end
        if (macierz(j,i)==10) plot(j,i,'x-g'); end
        if (macierz(j,i)==11) plot(j,i,'x-b'); end
        if (macierz(j,i)==12) plot(j,i,'x-m'); end
    end
end
for i=1:4,
    hold on;
    plot(centra(i,1),centra(i,2),'x-k');
end
% obliczenie nowych pozycji center
c1_x=0; c1_y=0; c1_l=0; c2_x=0; c2_y=0; c2_l=0; c3_x=0; c3_y=0; c3_l=0; c4_x=0; c4_y=0; c4_l=0;
for i=1:64
    if (tab(i,1)~=0)
        if (tab(i,3)==1), c1_x=c1_x+tab(i,2); c1_y=c1_y+tab(i,1); c1_l=c1_l+1; end
        if (tab(i,3)==2), c2_x=c2_x+tab(i,2); c2_y=c2_y+tab(i,1); c2_l=c2_l+1; end
        if (tab(i,3)==3), c3_x=c3_x+tab(i,2); c3_y=c3_y+tab(i,1); c3_l=c3_l+1; end
        if (tab(i,3)==4), c4_x=c4_x+tab(i,2); c4_y=c4_y+tab(i,1); c4_l=c4_l+1; end
    end
end
centra(1,2) = round(c1_x/c1_l); centra(1,1) = round(c1_y/c1_l);
centra(2,2) = round(c2_x/c2_l); centra(2,1) = round(c2_y/c2_l);
centra(3,2) = round(c3_x/c3_l); centra(3,1) = round(c3_y/c3_l);
centra(4,2) = round(c4_x/c4_l); centra(4,1) = round(c4_y/c4_l);
pause(2);
end
```

Wyniki działania programu przedstawiamy na następnej stronie. Rysunek 1 zawiera pozycje skupisk i centrów przed algorytmem, natomiast rysunek 2 po 6 kroku algorytmu. Czarne krzyżyki to centra, natomiast krzyżyki kolorowe odpowiadają środkom każdego

skupiska i zostały obliczone na samym początku programu. Nie są one wymagane do realizacji zadania, jednak w ten sposób widzimy gdzie jest środek skupiska i gdzie powinno znaleźć się centrum.



Rys.1 Przed rozpoczęciem pracy algorytmu k-średnich



Rys.2 Po 6 kroku algorytmu k-średnich

Jak widać nie wszystkie centra zostały przypisane do odpowiednich skupisk. Jedno z centrów waha się pomiędzy skupiskiem „zielonym” a „różowym”, wynika to z podobnej liczebności skupisk i przybliżonej odległości od każdego ze skupisk. W wyniku tego to centrum wpadło w pętlę w której bardzo wolno będzie zmieniało swoje położenie lub w skrajnym przypadku wcale nie zmieni swojej pozycji.

3. Klasyfikowanie znaków alfabetu przy pomocy metody k-średnich.

W tej części zadania postaramy się użyć metody off-line algorytmu k-średnich do zaklasyfikowania znaków alfabetów z poprzednich zajęć. W tym celu przygotujemy skrypt w środowisku MATLAB. Pierwsza część będzie polegała na uczeniu. Do dyspozycji mamy aż 116 różnych alfabetów a w każdym po 26 liter. Dlatego przyjmujemy 26 center które zostaną wylosowane spośród wszystkich dostępnych znaków. Kod odpowiedzialny za to zadanie zamieszczamy poniżej.

```
%algorytm uczenia
path = '.literki\';
tab_danych = [];
alfabet = [];
%pętla przeskakiwania po kolejnych znakach alfabetu
for k=0:25
    sign = char('a' + k);
    dir = strcat(path,sign,'\');
    %pętla przeskakiwania po kolejnych modyfikacjach danego znaku
    for i=1:110,
        path = '.literki\';
        name = strcat(sign,'_00');
        if i>9 name=strcat(sign,'_0'); end
        if i>99 name=strcat(sign,'_'); end
        %złożenie nazwy pliku
        name=strcat(name,int2str(i),'.bmp');
        %wczytanie danych z pliku
        full_path = strcat(dir,name);
        obraz = imread(full_path);
        tab_danych = [tab_danych;obraz(:)'];
        alfabet = [alfabet;sign];
    end
end
```

Kolejnym krokiem było wykonanie skryptu, który będzie obliczał do której grupy zaklasyfikować każde centrum. Klasyfikacja nastąpi na podstawie najmniejszej odległości. Jako miarę odległości zastosowaliśmy tak samo jak na poprzednich zajęciach kwadrat odległości euklidesowej. Dzięki temu porównamy skuteczność zaproponowanego rozwiązania. Kod skryptu zamieszczamy poniżej.

```
path='.literki\';
centra = zeros(26,576);
%petla do obrobki nazwy i sciezki do plików z danymi
for i=1:26,
    sign = char('a'+round(rand(1)*25));
    nr_alf = round(rand(1)*116);
    nr_alf_znak = int2str(nr_alf);
    if nr_alf<10,
        nr_alf_znak = strcat('00',nr_alf_znak);
    elseif nr_alf<100,
```

```
nr_alf_znak = strcat('0',nr_alf_znak);
end
dir = strcat(path,sign,'\');
name = strcat(sign,'_');
name = strcat(name,nr_alf_znak,'.bmp');
full_path = strcat(dir,name);
obraz = imread(full_path);
centra(i,:) = obraz(:);
end
%petla przesuwania centrów
while(1)
    obrazy = zeros(26,576);
    w_danych = zeros(1,length(tab_danych));
    for i=1:length(tab_danych),
        for j=1:26, obrazy(j,:)=tab_danych(i,:); end
        %zastosowanie kwadratowej odległości euklidesowej
        odleglosci = sum(((obrazy-centra).^2));
        %posortowanie danych
        [dane,index] = min(odleglosci);
        w_danych(i) = index;
    end
    %zapamiętanie danych centrow do następnej petli
    poprzednie_c = centra;
    for i=1:26,
        ktore = find(w_danych==i);
        ilosc = length(ktore);
        if ilosc>0,
            centra(i,:)=round(sum(tab_danych(ktore,:))./ilosc);
        end
    end
    %zastosowanie warunków stopu
    zm = sum(sum(abs(poprzednie_c-centra)));
    if licznik>=70, break; end
    if zm<1, break; end
end
%wczytanie zbioru znaków do badania
znaki = zeros(26,576);
for i=1:26,
    sign=char('a'+i-1);
    nr_alf_znak = strcat('00',int2str(2));
    %ustalenie katalogu i nazwy
    dir = strcat(path,sign,'\');
    name = strcat(name,'_',nr_alf_znak,'.bmp');
    full_path= strcat(dir,name);
    obrazy = imread(full_path);
    znaki(i,:) = obrazy(:);
end
obrazy=zeros(26,576);
litery=zeros(1,26);
for i=1:26, for j=1:26, obrazy(j,:)=centra(i,:); end
    %zastosowanie kwadratowej odległości euklidesowej
    odleglosci = sum(((obrazy-centra).^2));
    %posortowanie danych
    [dane,index]=min(odleglosci);
    litery(i)=char('a'+index-1);
end
%przekształcenie danych do znaków
litery=char(litery);
```

Na koniec wystarczyło przetestować ile znaków zostało rozpoznanych poprawnie a ile błędnie.

```

%petla badania skuteczności klasyfikatora
sukces=0; porazka=0;
for i=1:length(tab_danych),
    %przepisanie liter do zmiennej obrazu
    for j=1:26
        obrazy(j,:)=tab_danych(i,:);
    end
    %zastosowanie kwadratowej odległości euklidesowej
    odleglosci = sum(((obrazy-centra).^2));
    %posortowanie danych
    [dane,index]=min(odleglosci);
    %czy sukces czy porazka?
    if litery(index)==alfabet(i),
        sukces = sukces+1;
    else
        porazka = porazka+1;
    end
end
end

```

4. Zestawienie wyników algorytmu KNN i k-średnich.

Tabela 1 przedstawia wyniki działania algorytmu k-najbliższych sąsiadów.

Ilość sąsiadów	Poprawnie rozpoznanych znaków	Niepoprawnie rozpoznanych znaków	Wynik procentowy poprawnie rozpoznanych znaków
1	789	211	78,9 %
3	790	210	79 %
5	786	214	78,6 %
15	774	226	77,4 %
31	716	284	71,6 %
51	647	353	64,7 %

Tabela 1. Wyniki algorytmu KNN

Tabela 2 przedstawia wyniki działania algorytmu k-średnich dla tego samego alfabetu.

Poprawnie rozpoznanych znaków	Niepoprawnie rozpoznanych znaków	Wynik procentowy poprawnie rozpoznanych znaków
525	2335	18,36 %

Tabela 2. Wyniki algorytmu k-średnich

5. WNIOSKI

Jak widać z przeprowadzonych badań algorytm k-średnich wypadł bardzo słabo na tle algorytmu k-najbliższych sąsiadów. Ilość prawidłowo rozpoznanych znaków na poziomie niecałych 19% to bardzo słaby wynik. Jeśli porównywać wymagania pamięciowe obu algorytmów to metoda k-średnich wypada znacznie lepiej. Metoda k-średnich nie ma potrzeby zapamiętania tak dużej ilości wzorców uczących jak to miało miejsce w metodzie k-najbliższych sąsiadów. Co do wymagań systemowych odnośnie obliczeń, oba algorytmy wydają się być porównywalne i zużywają prawie tyle samo czasu. Błąd rozpoznawania w metodzie k-średnich polega na tym iż do niektórych skupisk zostały przydzielone więcej niż jedno centrum, podczas gdy do innych nie zostało przydzielone żadne centrum. Taka klasyfikacja jest wtedy obarczona znacznym błędem.